1 Purpose

Nag_dtbtrs (f07vec) solves a real triangular band system of linear equations with multiple right-hand sides, \( AX = B \) or \( A^T X = B \).

2 Specification

```c
#include <nag.h>
#include <nagf07.h>

void nag_dtbtrs (Nag_OrderType order, Nag_UploType uplo,
                    Nag_TransType trans, Nag_DiagType diag, Integer n, Integer kd,
                    Integer nrhs, const double ab[], Integer pdab, double b[], Integer pdb,
                    NagError *fail)
```

3 Description

Nag_dtbtrs (f07vec) solves a real triangular band system of linear equations \( AX = B \) or \( A^T X = B \).

4 References


5 Arguments

1: order – Nag_OrderType
   
   *Input*

   On entry: the order argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by order = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: uplo – Nag_UploType
   
   *Input*

   On entry: specifies whether \( A \) is upper or lower triangular.

   uplo = Nag_Upper
   \( A \) is upper triangular.

   uplo = Nag_Lower
   \( A \) is lower triangular.

   Constraint: uplo = Nag_Upper or Nag_Lower.

3: trans – Nag_TransType
   
   *Input*

   On entry: indicates the form of the equations.

   trans = Nag_NoTrans
   The equations are of the form \( AX = B \).
The equations are of the form $A^T X = B$.

**Constraint:** $\text{trans} = \text{Nag}_\text{NoTrans}, \text{Nag}_\text{Trans}$ or $\text{Nag}_\text{ConjTrans}$.

4: \hspace{1em} $\text{diag} = \text{Nag}_\text{DiagType}$ \hspace{1em} \text{Input}

**On entry:** indicates whether $A$ is a nonunit or unit triangular matrix.

$\text{diag} = \text{Nag}_\text{NonUnitDiag}$

$A$ is a nonunit triangular matrix.

$\text{diag} = \text{Nag}_\text{UnitDiag}$

$A$ is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

**Constraint:** $\text{diag} = \text{Nag}_\text{NonUnitDiag}$ or $\text{Nag}_\text{UnitDiag}$.

5: \hspace{1em} $n$ – Integer \hspace{1em} \text{Input}

**On entry:** $n$, the order of the matrix $A$.

**Constraint:** $n \geq 0$.

6: \hspace{1em} $kd$ – Integer \hspace{1em} \text{Input}

**On entry:** $kd$, the number of superdiagonals of the matrix $A$ if $\text{uplo} = \text{Nag}_\text{Upper}$, or the number of subdiagonals if $\text{uplo} = \text{Nag}_\text{Lower}$.

**Constraint:** $kd \geq 0$.

7: \hspace{1em} $\text{nrhs}$ – Integer \hspace{1em} \text{Input}

**On entry:** $r$, the number of right-hand sides.

**Constraint:** $\text{nrhs} \geq 0$.

8: \hspace{1em} $\text{ab}[\text{dim}]$ – const double \hspace{1em} \text{Input}

**Note:** the dimension, $\text{dim}$, of the array $\text{ab}$ must be at least $\text{max}(1, \text{pdab} \times n)$.

**On entry:** the $n$ by $n$ triangular band matrix $A$.

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of $A_{ij}$, depends on the $\text{order}$ and $\text{uplo}$ arguments as follows:

- If $\text{order} = \text{Nag}_\text{ColMajor}$ and $\text{uplo} = \text{Nag}_\text{Upper}$,
  
  $A_{ij}$ is stored in $\text{ab}[kd + i - j + (j - 1) \times \text{pdab}]$, for $j = 1, \ldots, n$ and $i = \text{max}(1, j - k_d), \ldots, j$;

- If $\text{order} = \text{Nag}_\text{ColMajor}$ and $\text{uplo} = \text{Nag}_\text{Lower}$,
  
  $A_{ij}$ is stored in $\text{ab}[i - j + (j - 1) \times \text{pdab}]$, for $j = 1, \ldots, n$ and $i = j, \ldots, \text{min}(n, j + k_d)$;

- If $\text{order} = \text{Nag}_\text{RowMajor}$ and $\text{uplo} = \text{Nag}_\text{Upper}$,
  
  $A_{ij}$ is stored in $\text{ab}[j - i + (i - 1) \times \text{pdab}]$, for $i = 1, \ldots, n$ and $j = i, \ldots, \text{min}(n, i + k_d)$;

- If $\text{order} = \text{Nag}_\text{RowMajor}$ and $\text{uplo} = \text{Nag}_\text{Lower}$,
  
  $A_{ij}$ is stored in $\text{ab}[kd + j - i + (i - 1) \times \text{pdab}]$, for $i = 1, \ldots, n$ and $j = \text{max}(1, i - k_d), \ldots, i$.

If $\text{diag} = \text{Nag}_\text{UnitDiag}$, the diagonal elements of $AB$ are assumed to be 1, and are not referenced.
9:  **pdab** – Integer

   *Input*

   *On entry:* the stride separating row or column elements (depending on the value of *order*) of the matrix \( A \) in the array \( ab \).

   *Constraint:* \( pdab \geq kd + 1 \).

10:  **b**[**dim**] – double

   *Input/Output*

   *Note:* the dimension, \( dim \), of the array \( b \) must be at least

   \[
   \max(1, \text{pdb} \times \text{nrhs}) \quad \text{when} \quad \text{order} = \text{Nag-ColMajor};
   \]

   \[
   \max(1, n \times \text{pdb}) \quad \text{when} \quad \text{order} = \text{Nag-RowMajor}.
   \]

   The \((i, j)\)th element of the matrix \( B \) is stored in

   \[
   b[(j - 1) \times \text{pdb} + i - 1] \quad \text{when} \quad \text{order} = \text{Nag-ColMajor};
   \]

   \[
   b[(i - 1) \times \text{pdb} + j - 1] \quad \text{when} \quad \text{order} = \text{Nag-RowMajor}.
   \]

   *On entry:* the \( n \) by \( r \) right-hand side matrix \( B \).

   *On exit:* the \( n \) by \( r \) solution matrix \( X \).

11:  **pdb** – Integer

   *Input*

   *On entry:* the stride separating row or column elements (depending on the value of *order*) in the array \( b \).

   *Constraints:*

   - if *order* = Nag-ColMajor, \( pdb \geq \max(1, n) \);
   - if *order* = Nag-RowMajor, \( pdb \geq \max(1, nrhs) \).

12:  **fail** – NagError *

   *Input/Output*

   The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument \( \langle value \rangle \) had an illegal value.

**NE_INT**

On entry, \( kd = \langle value \rangle \).

*Constraint:* \( kd \geq 0 \).

On entry, \( n = \langle value \rangle \).

*Constraint:* \( n \geq 0 \).

On entry, \( nrhs = \langle value \rangle \).

*Constraint:* \( nrhs \geq 0 \).

On entry, \( pdab = \langle value \rangle \).

*Constraint:* \( pdab > 0 \).

On entry, \( pdb = \langle value \rangle \).

*Constraint:* \( pdb > 0 \).
NE_INT_2
On entry, pdab = \langle value \rangle and kd = \langle value \rangle.
Constraint: pdab \geq kd + 1.
On entry, pdb = \langle value \rangle and n = \langle value \rangle.
Constraint: pdb \geq \max(1, n).
On entry, pdb = \langle value \rangle and nrhs = \langle value \rangle.
Constraint: pdb \geq \max(1, nrhs).

NE_INTERNAL_ERROR
An internal error has occurred in this function. Check the function call and any array sizes. If the
call is correct then please contact NAG for assistance.
An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_SINGULAR
Element \langle value \rangle of the diagonal is exactly zero. A is singular and the solution has not been
computed.

7 Accuracy
The solutions of triangular systems of equations are usually computed to high accuracy. See Higham
(1989).
For each right-hand side vector b, the computed solution x is the exact solution of a perturbed system of
equations \((A + E)x = b\), where
\[ |E| \leq c(k)\epsilon|A|, \]
c(k) is a modest linear function of k, and \(\epsilon\) is the machine precision.
If \(\hat{x}\) is the true solution, then the computed solution x satisfies a forward error bound of the form
\[ \frac{\|x - \hat{x}\|\infty}{\|x\|\infty} \leq c(k) \text{cond}(A, x)\epsilon, \quad \text{provided} \quad c(k) \text{cond}(A, x)\epsilon < 1, \]
where\(\text{cond}(A, x) = \frac{\|A^{-1}\|\|A\|\|x\|\infty}{\|x\|\infty}\).
Note that \(\text{cond}(A, x) \leq \text{cond}(A) = \|A^{-1}\|\|A\|\infty \leq \kappa_\infty(A)\); \(\text{cond}(A, x)\) can be much smaller than
\(\text{cond}(A)\) and it is also possible for \(\text{cond}(A^T)\) to be much larger (or smaller) than \(\text{cond}(A)\).
Forward and backward error bounds can be computed by calling nag_dtbtrs (f07vhc), and an estimate
for \(\kappa_\infty(A)\) can be obtained by calling nag_dtbcn (f07vgc) with \texttt{norm} = Nag conforme norm.

8 Parallelism and Performance
nag_dtbtrs (f07vvec) is threaded by NAG for parallel execution in multithreaded implementations of the
NAG Library.

nag_dtbtrs (f07vvec) makes calls to BLAS and/or LAPACK routines, which may be threaded within the
vendor library used by this implementation. Consult the documentation for the vendor library for further
information.
Please consult the X06 Chapter Introduction for information on how to control and interrogate the
OpenMP environment used within this function. Please also consult the Users’ Note for your
implementation for any additional implementation-specific information.
9 Further Comments

The total number of floating-point operations is approximately $2nk^2$ if $k \ll n$.

The complex analogue of this function is nag_ztbtrs (f07vsc).

10 Example

This example solves the system of equations $AX = B$, where

$$A = \begin{pmatrix} -4.16 & 0.00 & 0.00 & 0.00 \\ -2.25 & 4.78 & 0.00 & 0.00 \\ 0.00 & 5.86 & 6.32 & 0.00 \\ 0.00 & 0.00 & -4.82 & 0.16 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -16.64 & -4.16 \\ -13.78 & -16.59 \\ 13.10 & -4.94 \\ -14.14 & -9.96 \end{pmatrix}.$$ 

Here $A$ is treated as a lower triangular band matrix with one subdiagonal.

10.1 Program Text

/* nag_dtbtrs (f07vec) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * * Mark 7, 2001.
 */
#include <stdio.h>
#include <nag.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, k, kd, n, nrhs, pdab, pdb;
    Integer exit_status = 0;
    Nag_UploType uplo;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    double *ab = 0, *b = 0;
    #ifdef NAG_LOAD_FP
    /* The following line is needed to force the Microsoft linker 
     * to load floating point support */
    float force_loading_of_ms_float_support = 0;
    #endif /* NAG_LOAD_FP */
    #ifdef NAG_COLUMN_MAJOR
    #define AB_UPPER(I, J) ab[(J-1)*pdab + k + I - J - 1]
    #define AB_LOWER(I, J) ab[(I-1)*pdab + k + J - I - 1]
    #define B(I, J) b[(J-1)*pdb + I - 1]
    order = Nag_ColMajor;
    #else
    #define AB_UPPER(I, J) ab[(J-1)*pdab + k + I - J - 1]
    #define AB_LOWER(I, J) ab[(I-1)*pdab + k + J - I - 1]
    #define B(I, J) b[(J-1)*pdb + J - 1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    printf("nag_dtbtrs (f07vec) Example Program Results\\n\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    f07 – Linear Equations (LAPACK)
    f07vec.5
    Mark 25
    */
scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#endif
if (_WIN32)
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &kd, &nrhs);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &kd, &nrhs);
#endif
pdab = kd + 1;
#endif
#ifdef NAG_COLUMN_MAJOR
    pdb = n;
#else
    pdb = nrhs;
#endif
#ifdef _WIN32
    scanf_s("%39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s%*[\n] ", nag_enum_arg);
#endif
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
k = kd + 1;
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
#ifdef _WIN32
        scanf_s("%lf", &AB_UPPER(i, j));
#else
        scanf("%lf", &AB_UPPER(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#else
    for (i = 1; i <= n; ++i)
    {
#ifdef _WIN32
        scanf_s("%lf", &AB_LOWER(i, j));
#else
        scanf("%lf", &AB_LOWER(i, j));
#endif
    }
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
}
/* Read B from data file */
for (i = 1; i <= n; ++i)
{
    for (j = 1; j <= nrhs; ++j)
    {#ifdef _WIN32
        scanf_s("%lf", &B(i, j));
    #else
        scanf("%lf", &B(i, j));
    #endif
    #ifdef _WIN32
        scanf_s("%*[\n] ");
    #else
        scanf("%*[\n] ");
    #endif
}
/* Compute solution */
/* nag_dtbtrs (f07vec).  
 * Solution of real band triangular system of linear  
 * equations, multiple right-hand sides  
 */
nag_dtbtrs(order, uplo, Nag_NoTrans, Nag_NonUnitDiag, n, 
        kd, nrhs, ab, pdab, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dtbtrs (f07vec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
/* nag_gen_real_mat_print (x04cac).  
 * Print real general matrix (easy-to-use)  
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, 
        b, pdb, "Solution(s)", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ab);
NAG_FREE(b);
return exit_status;

10.2 Program Data

nag_dtbtrs (f07vec) Example Program Data
4 1 2 :Values of n, kd and nrhs
   Nag_Lower :Value of uplo
-4.16
-2.25 4.78
  5.86 6.32
     -4.82 0.16 :End of matrix A
-16.64 -4.16
-13.78 -16.59
  13.10 -4.94
### Program Results

**nag_dtbtrs (f07vec) Example Program Results**

<table>
<thead>
<tr>
<th>Solution(s)</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>-1.0000</td>
<td>-3.0000</td>
</tr>
<tr>
<td>3</td>
<td>3.0000</td>
<td>2.0000</td>
</tr>
<tr>
<td>4</td>
<td>2.0000</td>
<td>-2.0000</td>
</tr>
</tbody>
</table>